



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

<Name>

<Date>



# Outline

---

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

# Executive Summary

---

- Summary of methodologies

In data collection, two major method will be used: SpaceX REST API & website scraping.

- Summary of all results

# Introduction

---

- Project background and context

In this project, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

- Problems you want to find answers

The problems I want to find the answer is to determine if the first stage will land, we can determine the cost of a launch. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.



Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

# Data Collection

---

- Describe how data sets were collected.

Method 1 : Space X REST API

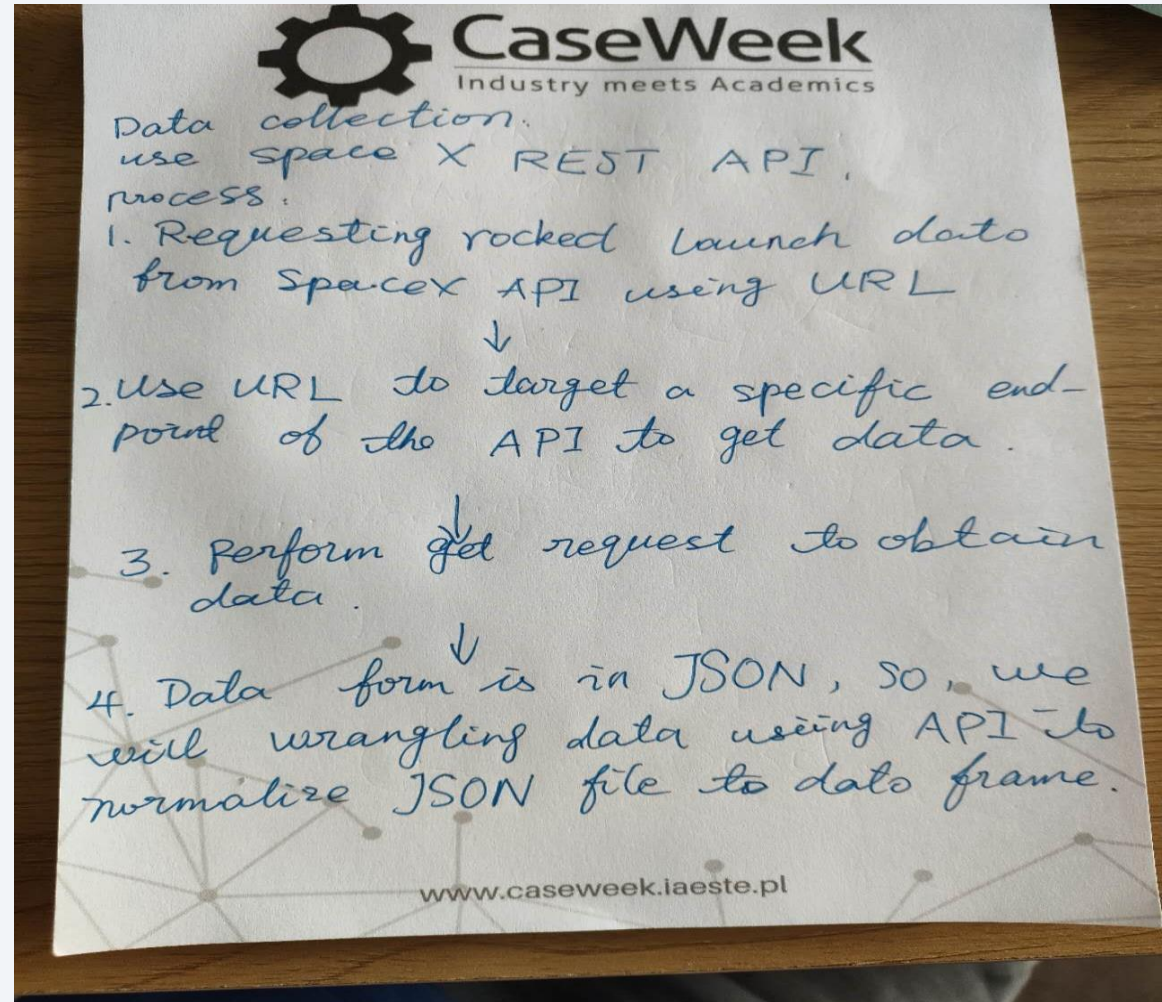
Requesting rocket data from SpaceX API using URL.

Method 2: Scraping data from Wikipedia

# Data Collection – SpaceX API

- General flowchart is shown as right figure.
- The GitHub URL of the completed SpaceX API calls notebook:

[https://github.com/Urysohnish/Data-Science/blob/main/final\\_pro\\_data\\_collecting.ipynb](https://github.com/Urysohnish/Data-Science/blob/main/final_pro_data_collecting.ipynb)





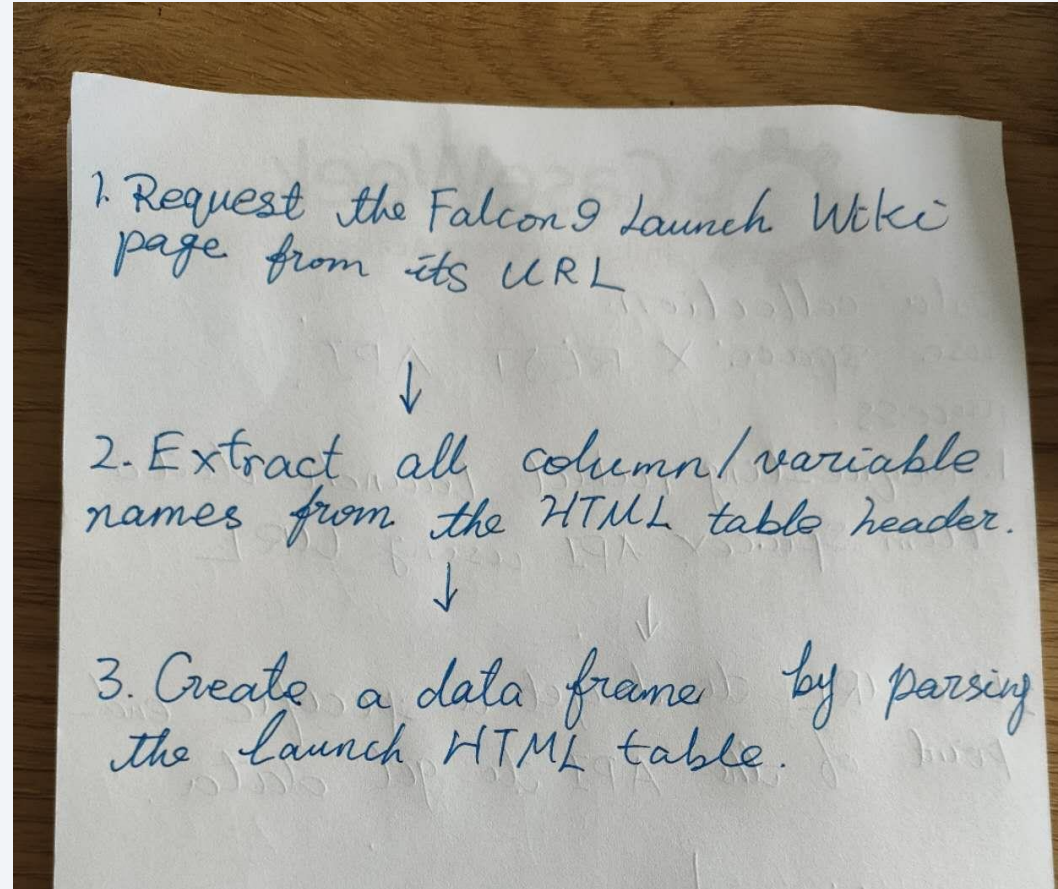
# Data Collection - Scraping

---

- General flowchart is shown as right figure.

- The GitHub URL of the completed web scraping

[https://github.com/Urysohnish/Data-Science/blob/main/final\\_pro\\_d/atacollection\\_scri.ipynb](https://github.com/Urysohnish/Data-Science/blob/main/final_pro_d/atacollection_scri.ipynb)



# Data Wrangling

---

- General process of Data Wrangling:
  1. Using `isnull().sum()` to fix the positions of missing data.
  2. Using appropriate method to deal each missing value, such as adding the missing values as the average of the nearby values.
  3. `.replace()` function to replace `np.nan` values in the data with the mean you calculated.
- The GitHub URL of completed data wrangling related notebooks, as an external reference and peer-review purpose:

[https://github.com/Urysohnish/Data-Science/blob/main/final\\_pro\\_data\\_collecting.ipynb](https://github.com/Urysohnish/Data-Science/blob/main/final_pro_data_collecting.ipynb)

# EDA with Data Visualization

---

- Summarize what charts were plotted and why you used those charts:

Bar chart

Line chart

Pie chart etc.

- The GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose:
- <https://github.com/Urysohnish/Data-Science/blob/main/final2.2.ipynb>

# EDA with SQL

---

- Using bullet point format, summarize the SQL queries you performed:

Select

Group by

Order by

Distinct() etc.

- The GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose:

<https://github.com/Urysohnish/Data-Science/blob/main/jupyter-labs-eda-sql-final2.4.ipynb>



# Build an Interactive Map with Folium

---

- In the map, I created and added o markers, circles, lines.
- The GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose:
- [https://github.com/Urysohnish/Data-Science/blob/main/IBM-DS0321EN-SkillsNetwork\\_labs\\_module\\_3\\_lab\\_jupyter\\_launch\\_site\\_location.jupyterlite.ipynb](https://github.com/Urysohnish/Data-Science/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb)

# Build a Dashboard with Plotly Dash

---

- The GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose:

[https://github.com/Urysohnish/Data-Science/blob/main/finalpro\\_notebook.py](https://github.com/Urysohnish/Data-Science/blob/main/finalpro_notebook.py)

# Predictive Analysis (Classification)

---

- Summarize how you built, evaluated, improved, and found the best performing classification model:

We split the data set into test set and training set. Built the different models and train the model on training set. Then get result on test set.

- The GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose:

<https://github.com/Urysohnish/Data-Science/blob/main/ai.ipynb>

# Results

---

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower half of the image. The overall effect is dynamic and technological.

Section 2

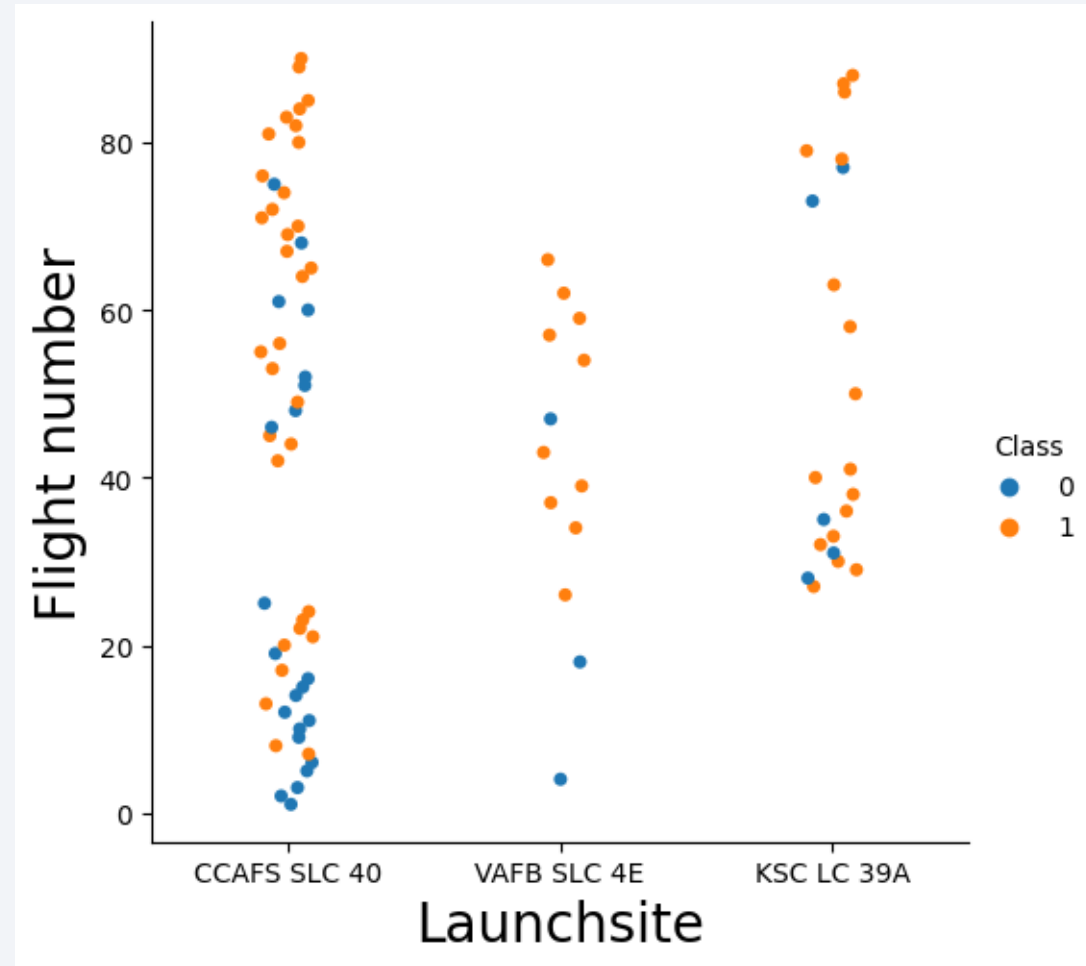
# Insights drawn from EDA



# Flight Number vs. Launch Site

- The scatter plot of Flight Number vs. Launch Site

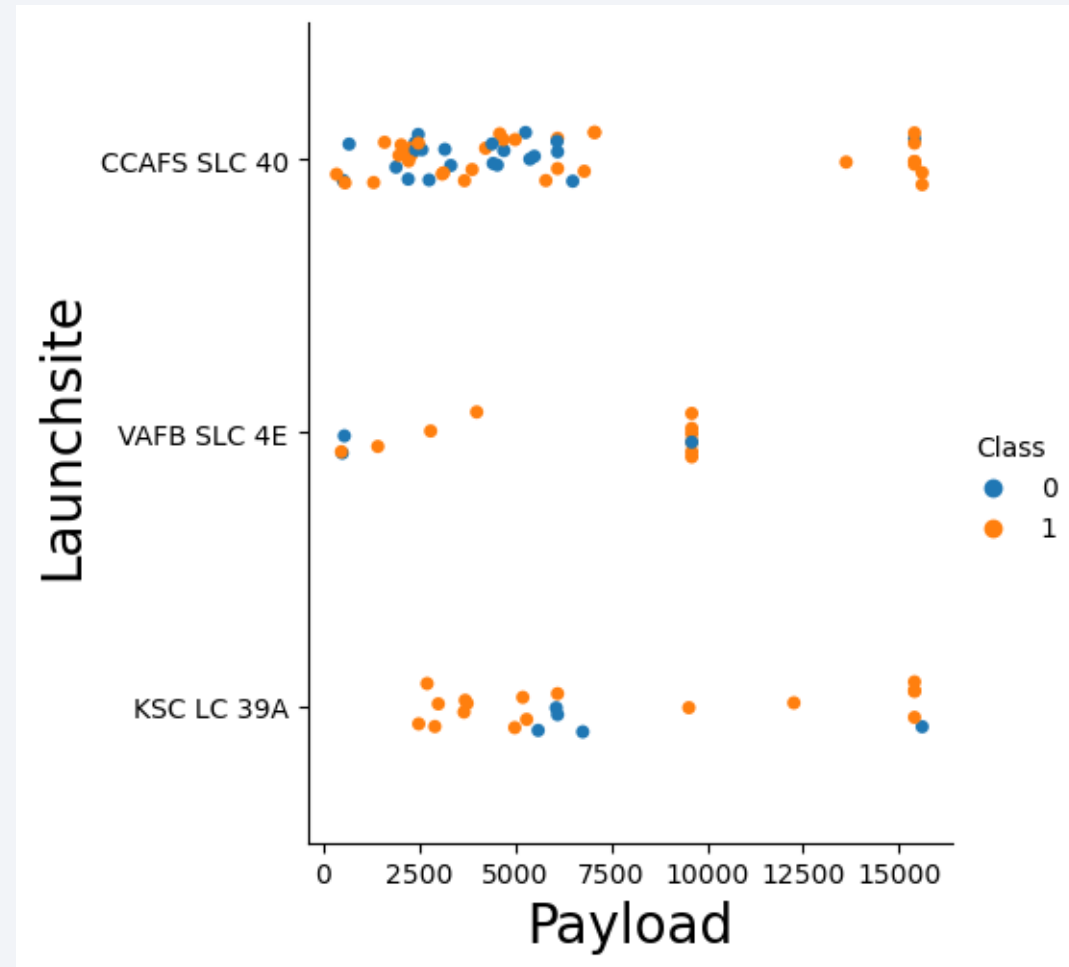
The totally launching number in launch site CCAFS SLC 40 is the greatest. The successful launching rate in launch site is the highest.



# Payload vs. Launch Site

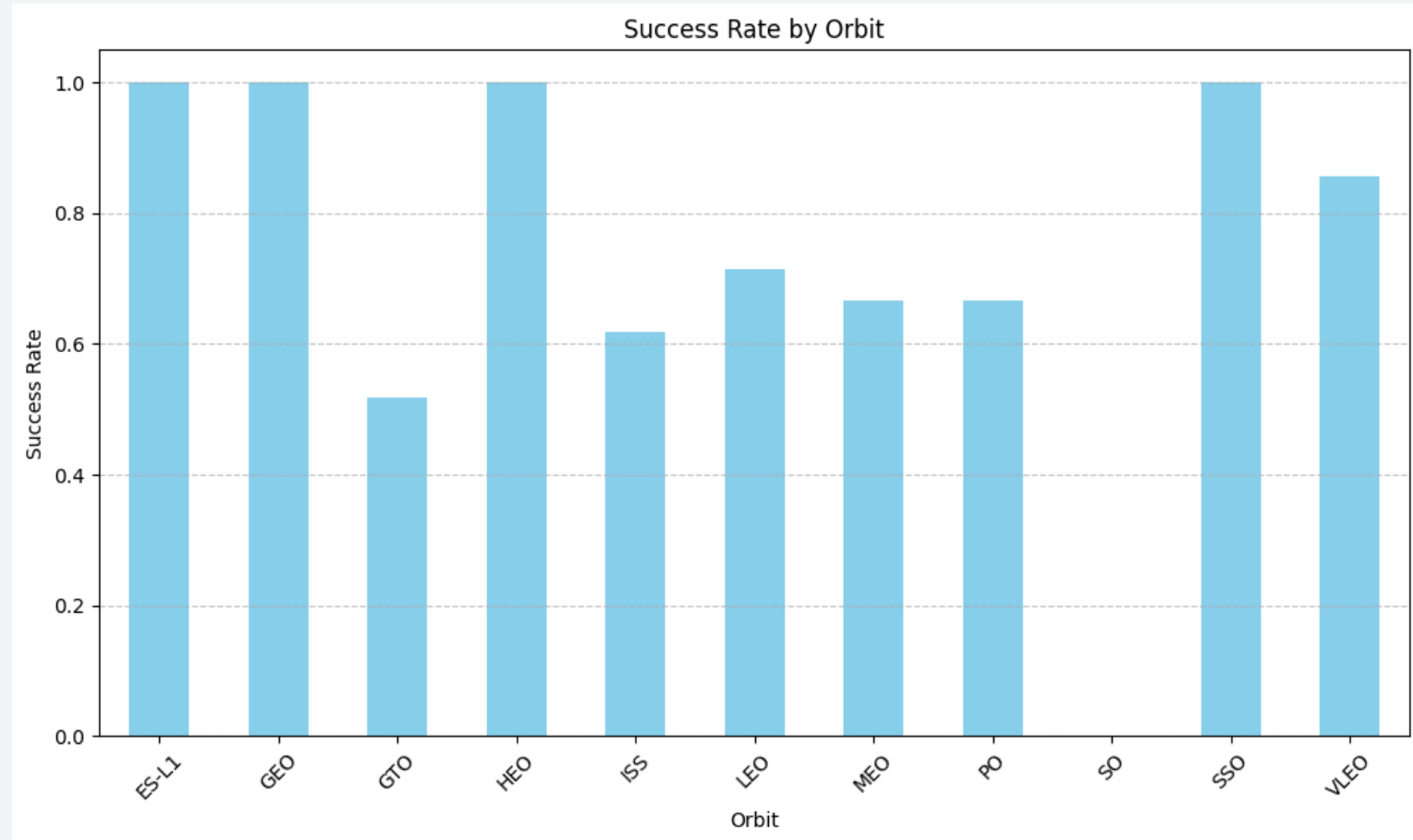
- The scatter plot of Payload vs. Launch Site

For payload above 10000 in launch site CCAFS SLC 40 the successful rate is approaching 100%.



# Success Rate vs. Orbit Type

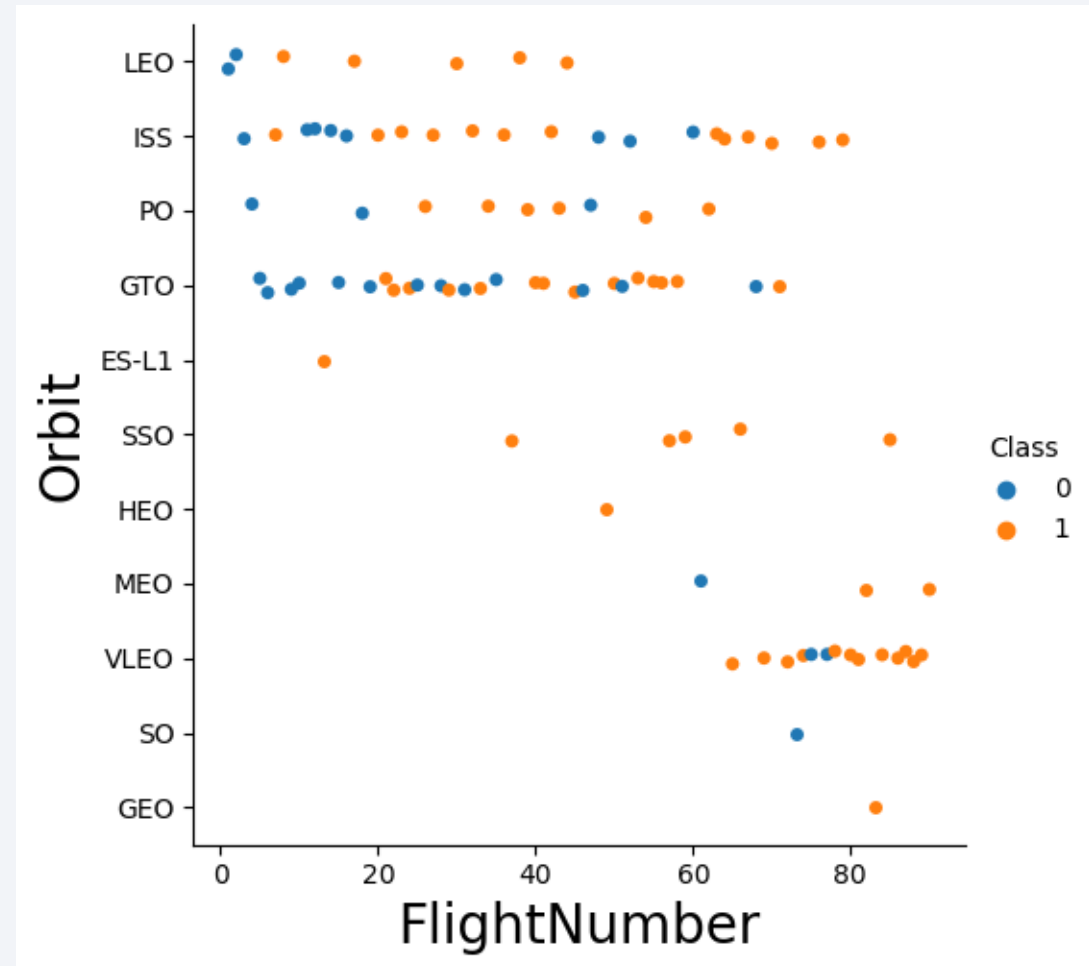
- The a bar chart for the success rate of each orbit type
- For four orbit type ES-LI, GEO, HEO, SSO, the successful launching rates are 100%.





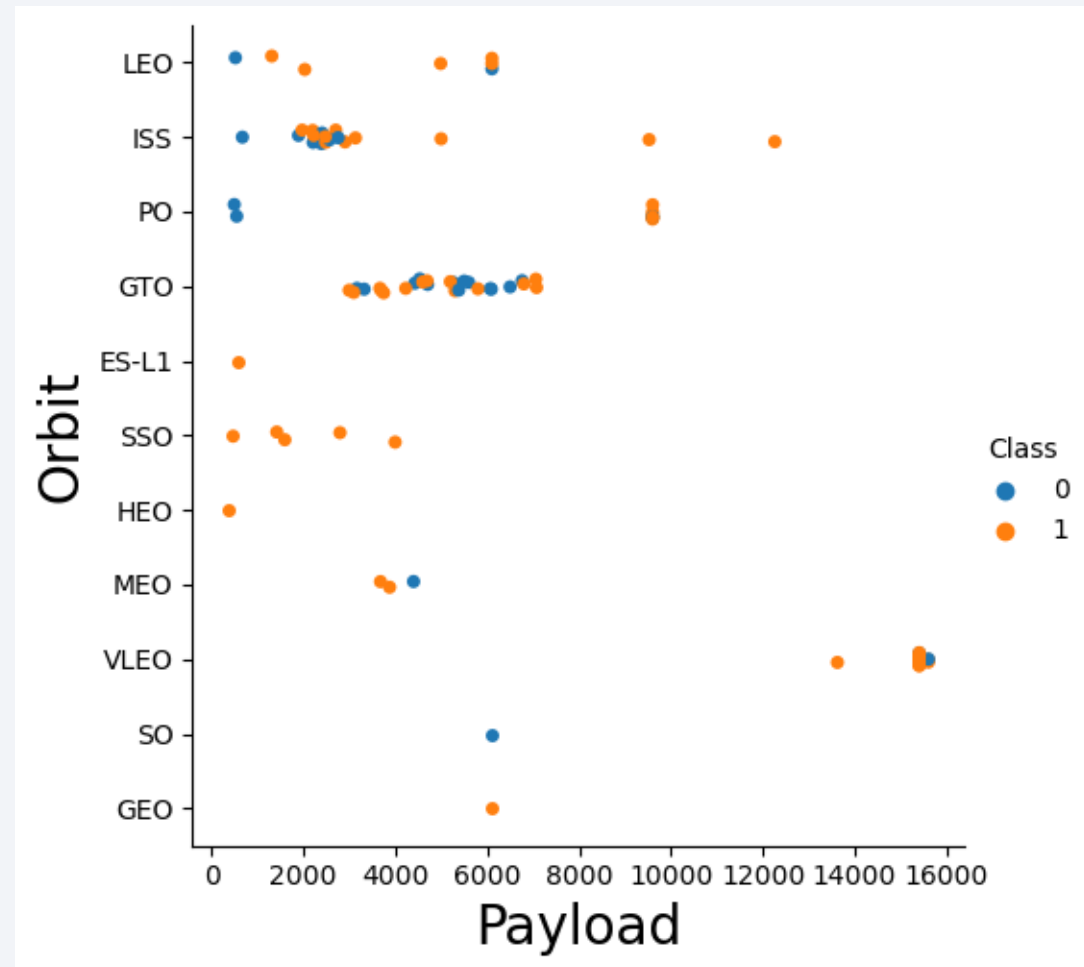
# Flight Number vs. Orbit Type

- The scatter point of Flight number vs. Orbit type
- Generally as the number of the flights increasing, there are more possible that the launching in each orbit is success.



# Payload vs. Orbit Type

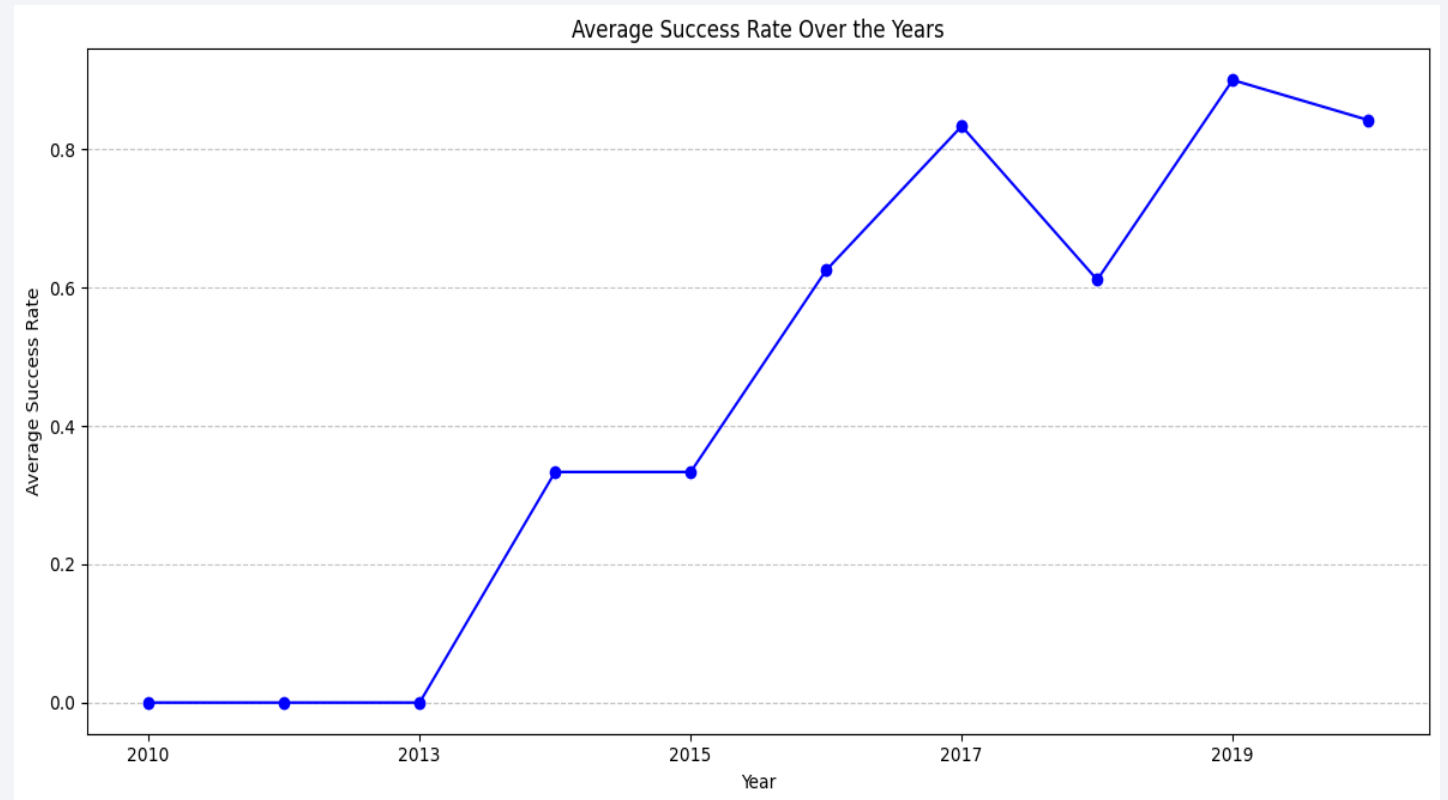
- The scatter point of payload vs. orbit type
- For orbit ES-L1, SSO, HEO, no matter what value the payload is the success launching rate is always 100%.



# Launch Success Yearly Trend

---

- The line chart of yearly average success rate
- Generally as the time increasing the success launching rate is raising, though there is a small fluctuation in year 2017-2019.



# All Launch Site Names

---

- Find the names of the unique launch sites

Display the names of the unique launch sites in the space mission

```
[15]: %sql select distinct(Launch_Site) from SPACEXTABLE
```

```
* sqlite:///my_data1.db
```

Done.

```
[15]: Launch_Site
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```



# Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with 'CCA'

```
In [24]: 1 %sql select * from SPACEXTABLE where LauNch_Site LIKE 'CCA%' limit 5;
```

\* sqlite:///my\_data1.db  
Done.

Out[24]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

# Total Payload Mass

---

- Calculate the total payload carried by boosters from NASA
- The total sum is 45596KG.

## Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
[47]: %sql select sum(PAYLOAD_MASS_KG_) from SPACEXTABLE where Customer = 'NASA (CRS)'
* sqlite:///my_data1.db
Done.
[47]: sum(PAYLOAD_MASS_KG_)
45596
```

# Average Payload Mass by F9 v1.1

---

- Calculate the average payload mass carried by booster version F9 v1.1

## Task 4

Display average payload mass carried by booster version F9 v1.1

```
[38]: %sql select avg(PAYLOAD_MASS_KG_) from SPACEXTABLE where Booster_version Like 'F9 V1.1%'
* sqlite:///my_data1.db
Done.
[38]: avg(PAYLOAD_MASS_KG_)
2534.6666666666665
```

# First Successful Ground Landing Date

---

- Find the dates of the first successful landing outcome on ground pad

```
Task 5
List the date when the first succesful landing outcome in ground pad was acheived.
Hint: Use min function

[39]: %sql select min(Date) from SPACEXTABLE where Mission_Outcome = 'Success'
* sqlite:///my_data1.db
Done.
[39]: min(Date)
2010-04-06
```

# Successful Drone Ship Landing with Payload between 4000 and 6000

---

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

## Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
[48]: here Landing_Outcome = 'Success (drone ship)' and (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

```
* sqlite:///my_data1.db
```

Done.

```
[48]: Booster_Version
```

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

# Total Number of Successful and Failure Mission Outcomes

---

- Calculate the total number of successful and failure mission outcomes

```
[41]: %sql select count(*) from SPACEXTABLE where Mission_Outcome = 'Success'
* sqlite:///my_data1.db
Done.

[41]: count(*)
-----
      98

[44]: %sql select count(*) from SPACEXTABLE where Mission_Outcome Like 'Failure%'
* sqlite:///my_data1.db
Done.

[44]: count(*)
-----
       1
```

# Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
[46]: %sql select booster_version from SPACEXTABLE where PAYLOAD_MASS_KG_ = (select max(PAYLOAD_MASS_KG_) from SPACEXTABLE)
* sqlite:///my_data1.db
Done.
```

[46]: **Booster\_Version**

F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6

LILI  
邀请你进行视频通话



# 2015 Launch Records

---

- List the failed landing\_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
Task 9

List the records which will display the month names, failure landing_outcomes in drone ship
,booster versions, launch_site for the months in year 2015.

Note: SQLite does not support monthnames. So you need to use substr(Date, 4,
2) as month to get the months and substr(Date,7,4)='2015' for year.

[56]: SPACEXTABLE WHERE substr(Date, 7, 4) = '2015' AND landing_outcome = 'Failure (drone ship)';

* sqlite:///my_data1.db
Done.

[56]: month Landing_Outcome Booster_Version Launch_Site
```

# Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

---

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
[54]: %sql select Landing_Outcome, COUNT(*) as count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' and '2017-03-20'
```

```
* sqlite:///my_data1.db  
Done.
```

```
[54]:
```

Landing_Outcome	count
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

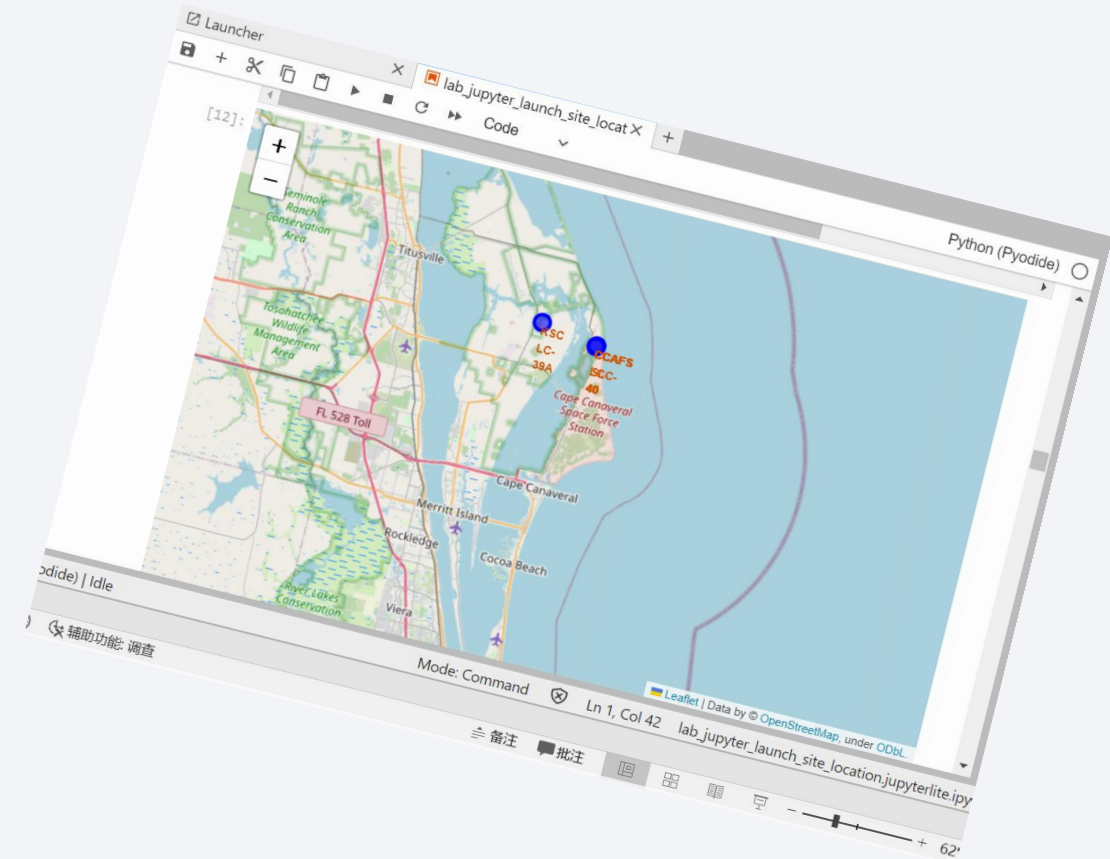
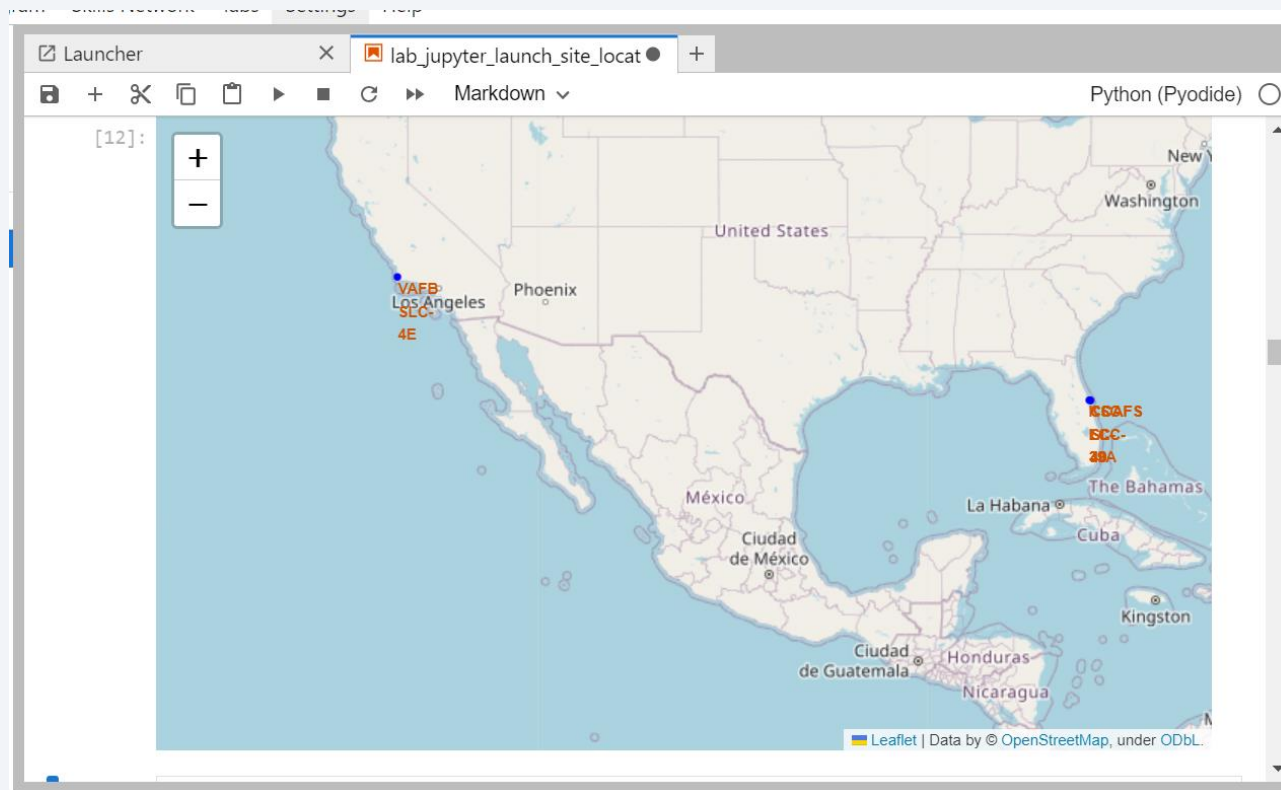
Reference Links

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

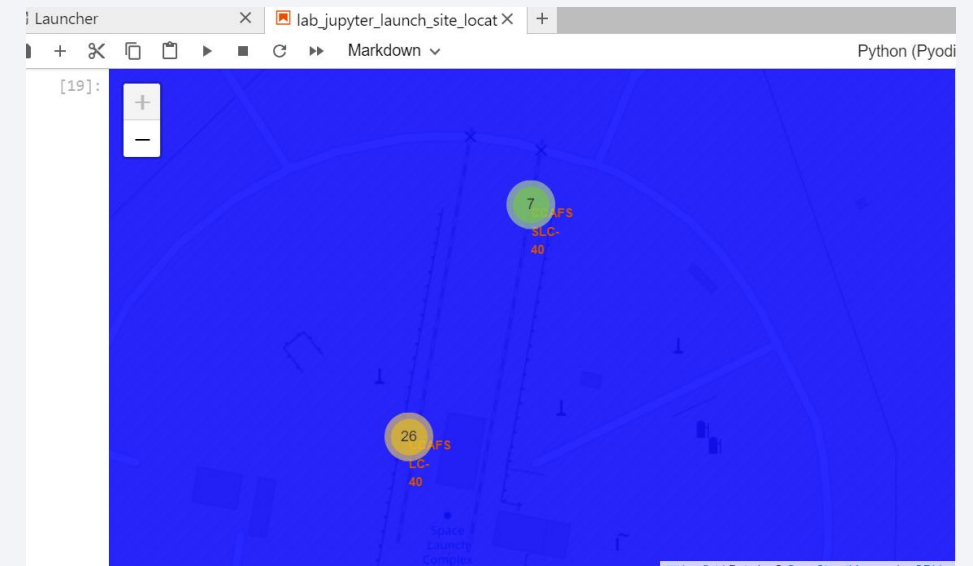
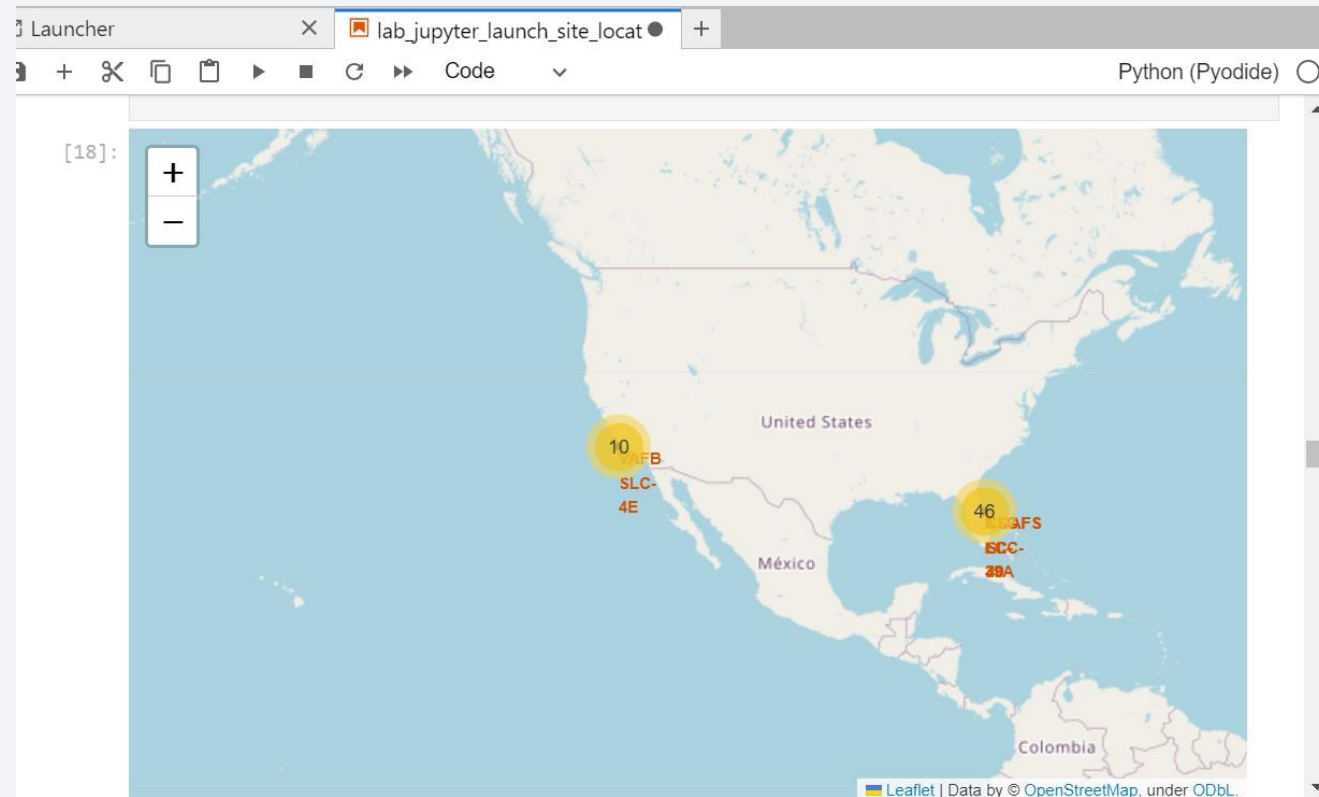
Section 3

# Launch Sites Proximities Analysis

# Mark all launch sites on a map

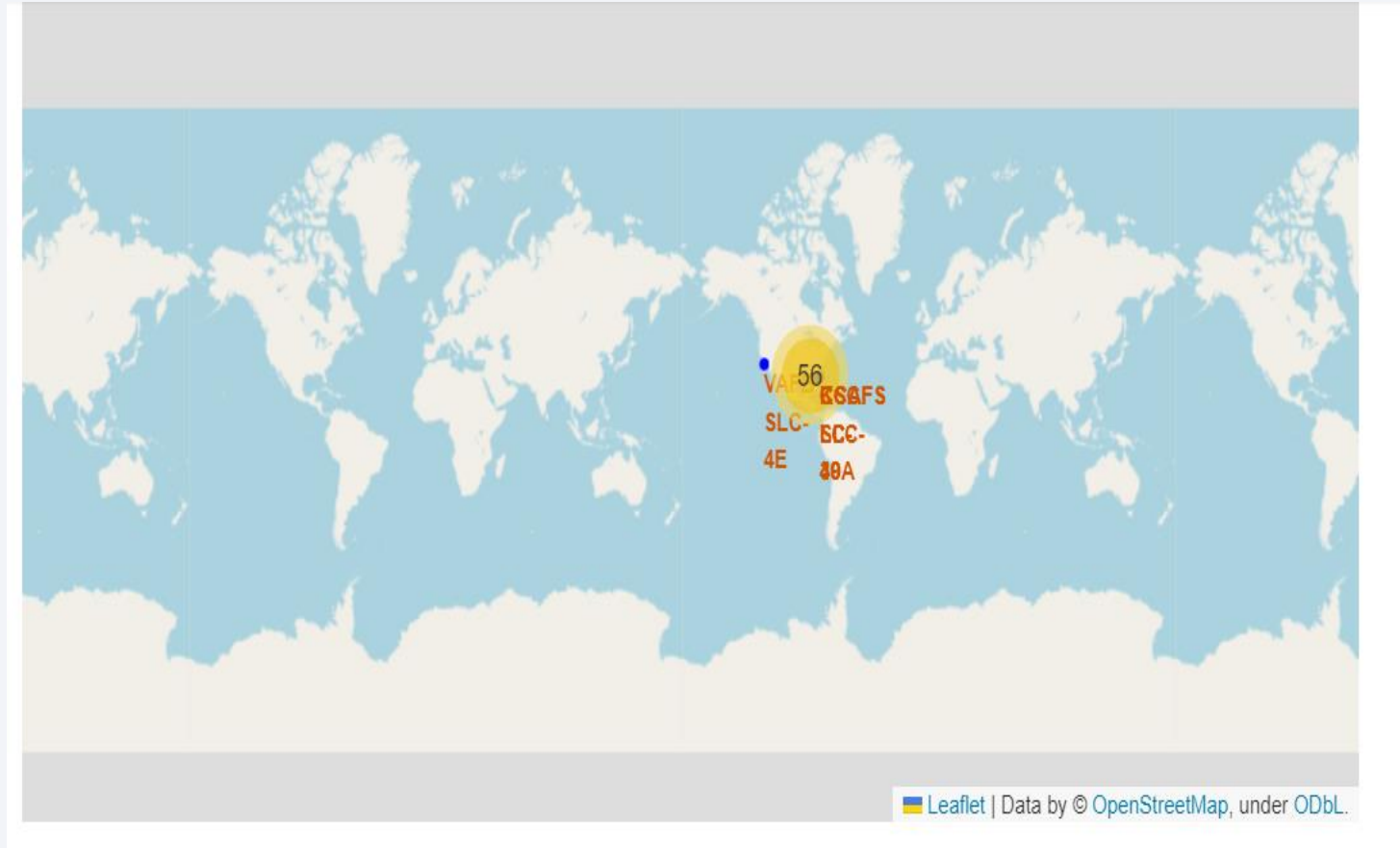


# Mark the success/failed launches for each site on the map



# Calculate the distances between a launch site to its proximities

---





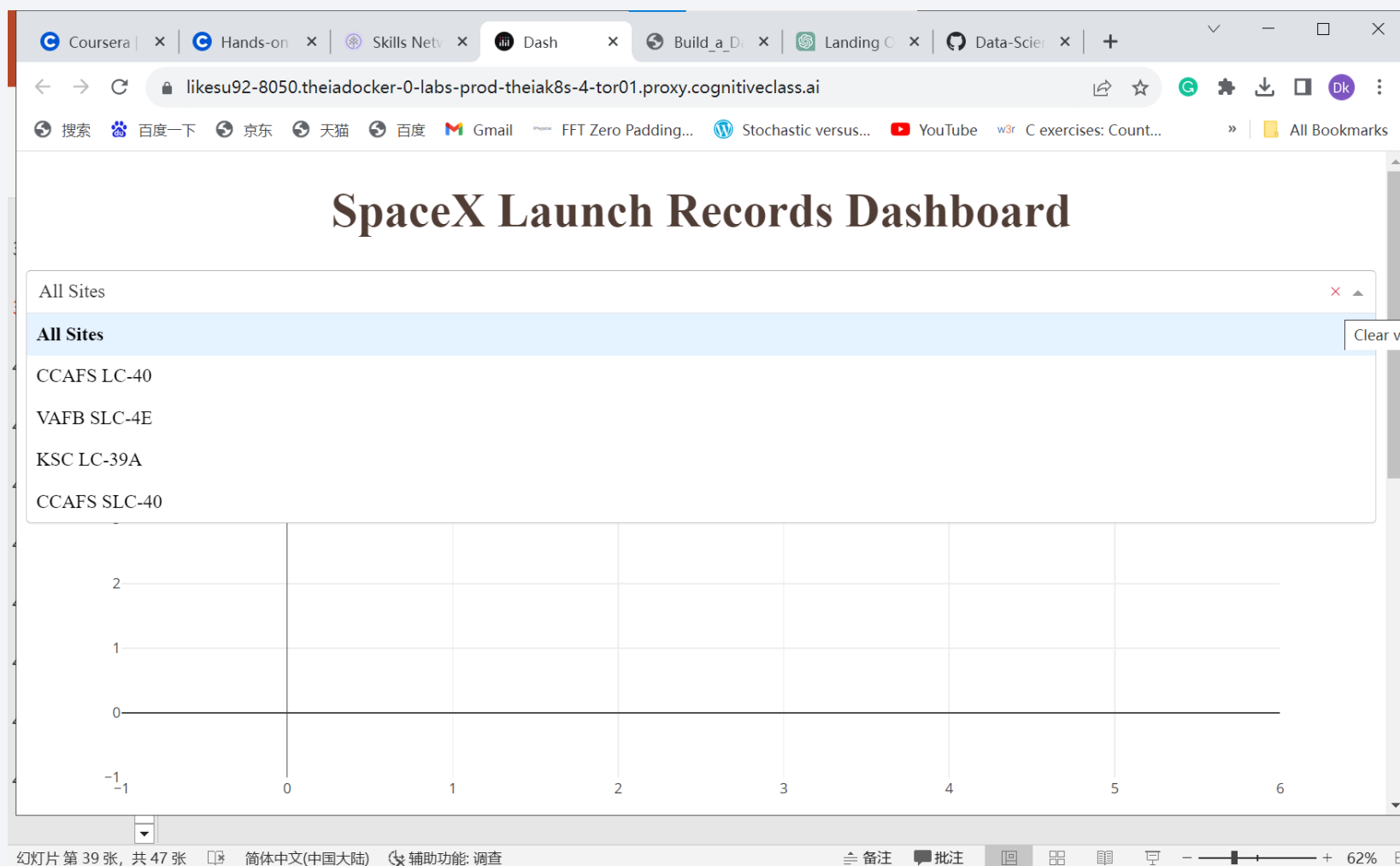


Section 4

# Build a Dashboard with Plotly Dash

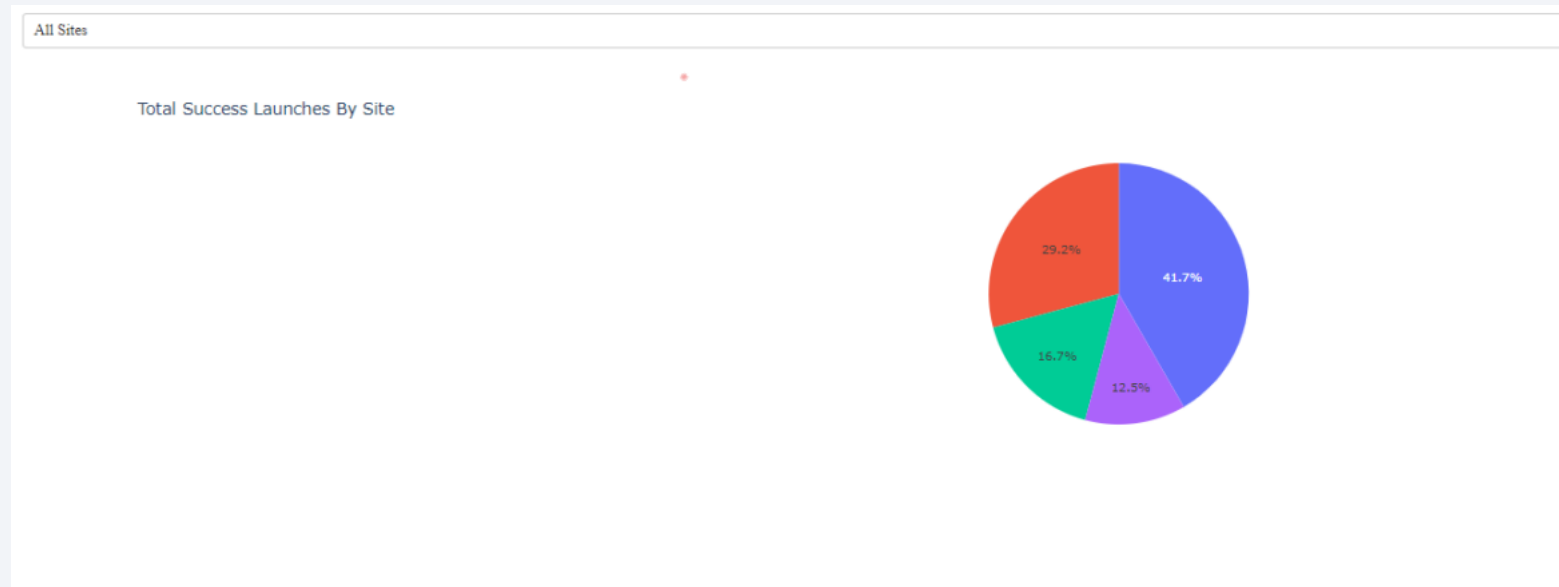


# Add a dropdown list to enable Launch Site selection



# Add a pie chart to show the total successful launches count for all sites

---

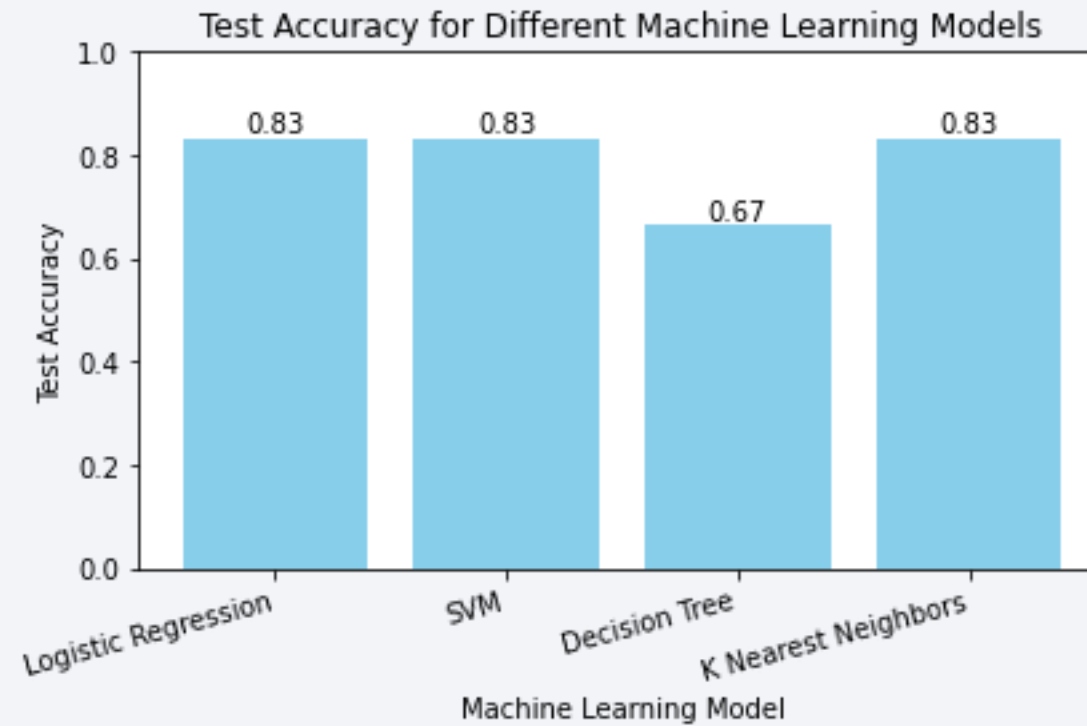


Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

---

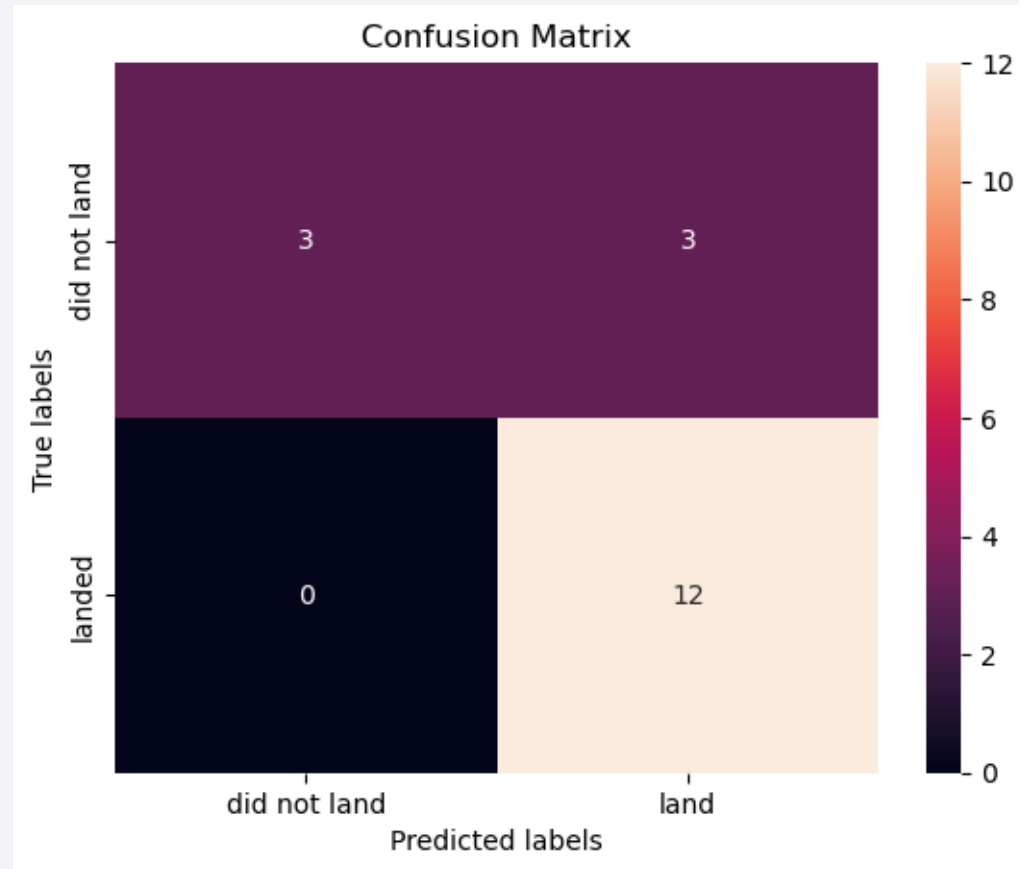


- The models Logistic Regression, SVM, K nearest neighbors have the highest classification accuracy.

# Confusion Matrix

---

The best performing model are K Nearest Neighbors , SVM and Logistic Regression, they all have the same accuracy and same confusion matrix.



# Conclusions

---

- Point 1

In these three top models, we could see the successful landing rate is very high, almost 80%.

- Point 2

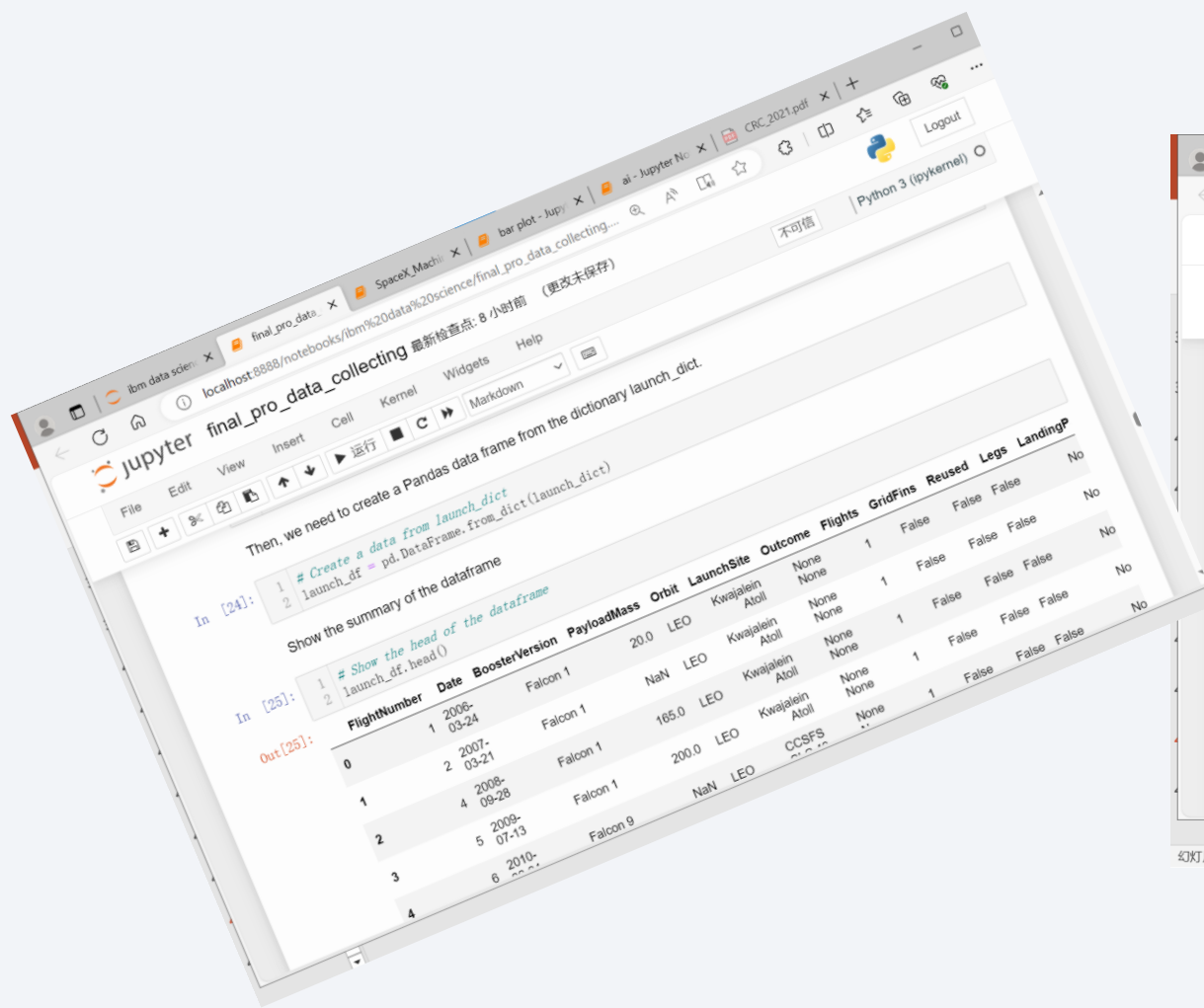
Successful landing rate is highly connected with the orbit type. Four orbit type ES-LI, GEO, HEO, SSO are among the perfect ones.

- Point 3

Successful landing rate is also highly connected with the place of submission. VAFB SLC 4E and KSC LC 39A have relatively high successful rate.



# Appendix



Then, we need to create a Pandas data frame from the dictionary `launch_dict`.

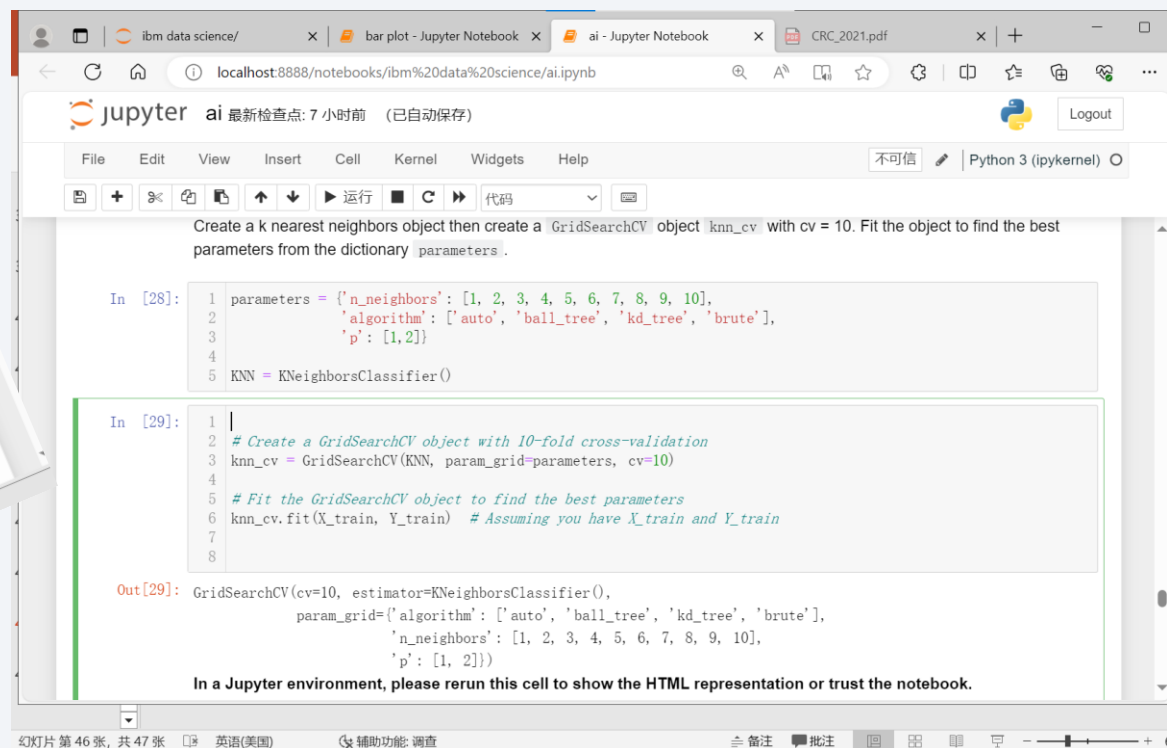
```
In [24]: 1 # Create a data from launch_dict
        2 launch_df = pd.DataFrame.from_dict(launch_dict)
```

Show the summary of the dataframe

```
In [25]: 1 # Show the head of the dataframe
        2 launch_df.head()
```

Out[25]:

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingP
0	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None	1	False	False	False	No
1	2007-03-21	Falcon 1	165.0	LEO	Kwajalein Atoll	None	1	False	False	False	No
2	2008-09-28	Falcon 1	200.0	LEO	Kwajalein Atoll	None	1	False	False	False	No
3	2009-07-13	Falcon 9	NaV	LEO	CCSFS	None	1	False	False	False	No
4	2010-01-10	Falcon 9	NaV	LEO	CCSFS	None	1	False	False	False	No



Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with `cv = 10`. Fit the object to find the best parameters from the dictionary `parameters`.

```
In [28]: 1 parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        2           'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
        3           'p': [1, 2]}
        4
        5 KNN = KNeighborsClassifier()
```

```
In [29]: 1
        2 # Create a GridSearchCV object with 10-fold cross-validation
        3 knn_cv = GridSearchCV(KNN, param_grid=parameters, cv=10)
        4
        5 # Fit the GridSearchCV object to find the best parameters
        6 knn_cv.fit(X_train, Y_train) # Assuming you have X_train and Y_train
        7
        8
```

Out[29]: `GridSearchCV(cv=10, estimator=KNeighborsClassifier(), param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'], 'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'p': [1, 2]})`

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

Thank you!

